dnsop   P. Wouters
Internet-Draft   Red Hat
Intended status: Experimental   February 18, 2016
Expires: August 21, 2016

                     Chain Query requests in DNS
                  draft-ietf-dnsop-edns-chain-query-07

Abstract

   This document defines an EDNS0 extension that can be used by a
   security-aware validating resolver configured to use a Forwarder to
   send a single query, requesting a complete validation path along with
   the regular query answer. The reduction in queries potentially
   lowers the latency and reduces the need to send multiple queries at
   once. This extension mandates the use of source IP verified
   transport such as TCP or UDP with EDNS-COOKIE so it cannot be abused
   in amplification attacks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Traditionally, a DNS client operates in stub-mode. For each DNS
   question the DNS client needs to resolve, it sends a single query to
   an upstream Recursive Resolver to obtain a single DNS answer. When
   DNSSEC [RFC4033] is deployed on such DNS clients, validation requires
   that the client obtains all the intermediate information from the DNS
   root down to the queried-for hostname so it can perform DNSSEC
   validation on the complete chain of trust.

Currently, applications send out many UDP requests concurrently.
This requires more resources on the DNS client with respect to state
(cpu, memory, battery) and bandwidth. There is also no guarantee
that the initial set of UDP questions will result in all the records
required for DNSSEC validation. More round trips could be required
depending on the resulting DNS answers. This especially affects
high-latency links.

This document specifies an EDNS0 extension that allows a validating
Resolver running as a Forwarder to open a TCP connection to another
Resolver and request a DNS chain answer using one DNS query/answer
pair. This reduces the number of round trips to two. If combined
with long lived TCP or [TCP-KEEPALIVE] there is only one round trip.
While the upstream Resolver still needs to perform all the individual
queries required for the complete answer, it usually has a much
bigger cache and does not experience significant slowdown from last-
mile latency.

This EDNS0 extension allows the Forwarder to indicate which part of
the DNS hierarchy it already contains in its cache. This reduces the
amount of data required to be transferred and reduces the work the
upstream Recursive Resolver has to perform.

This EDNS0 extension is only intended to be sent by Forwarders to
Recursive Resolvers. It MUST be ignored by Authoritative Servers.

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

2. Terminology

The DNS terminology used in this document is that of [RFC7719].
Additionally, the following terms are used:

Recursive Resolver: A nameserver that is responsible for resolving
   domain names for clients by following the domain's delegation
   chain, starting at the root. Recursive Resolvers frequently use
   caches to be able to respond to client queries quickly. Described
   in [RFC1035] chapter 7.

Validating Resolver: A recursive nameserver that also performs
   DNSSEC [RFC4033] validation. Also known as "security-aware
   resolver".

3. Overview

   When DNSSEC is deployed on a host, it can no longer delegate all DNS
   work to the upstream Recursive Resolver. Obtaining just the DNS
   answer itself is not enough to validate that answer using DNSSEC.
   For DNSSEC validation, the DNS client requires a locally running
   validating Resolver so it can confirm DNSSEC validation of all
   intermediary DNS answers. It can configure itself as a Forwarder if
   it obtains the IP addresses of one or more Recursive Resolvers that
   are available, or as a stand-alone Recursive Resolver if no
   functional Recursive Resolvers were obtained. Generating the
   required queries for validation adds a significant delay in answering
   the DNS question of the locally running application. The application
   must wait while the Resolver validates all intermediate answers.
   Each round-trip adds to the total time waiting on DNS resolution with
   validation to complete. This makes DNSSEC resolving impractical for
   devices on networks with a high latency.

   This document defines the CHAIN option that allows the Resolver to
   request all intermediate DNS data it requires to resolve and validate
   a particular DNS answer in a single round-trip. The Resolver could
   be part of the application or a Recursive Resolver running on the
   host.

   Servers answering with CHAIN data should ensure that the transport is
   TCP or source IP address verified UDP. See Section 8. This avoids
   abuse in DNS amplification attacks.

   Applications that support CHAIN internally can perform validation
   without requiring the host to run a Recursive Resolver. This is
   particularly useful for virtual servers in a cloud or container based
   deployment where it is undesirable to run a Recursive Resolver per
   virtual machine.

   The format of this option is described in Section 4.

   As described in Section 5.4, a Recursive Resolver could use this
   EDNS0 option to include additional data required by the Resolver in
   the Authority Section of the DNS answer packet when using a source IP
   verified transport. The Answer Section remains unchanged from a
   traditional DNS answer and contains the answer and related DNSSEC
   entries.

   An empty CHAIN EDNS0 option MAY be sent over any transport as a
   discovery method. A DNS server receiving such an empty CHAIN option
   SHOULD add an empty CHAIN option in its answer to indicate that it
   supports CHAIN for source IP address verified transports.

The mechanisms provided by CHAIN raise various security related
concerns, related to the additional work, bandwidth, amplification
attacks as well as privacy issues with the cache. These concerns are
described in Section 8.

4. Option Format

This draft uses an EDNS0 [RFC6891] option to include client IP
information in DNS messages. The option is structured as follows:

```
                        1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-------------------------------+-------------------------------+
  !          OPTION-CODE          !         OPTION-LENGTH         !
  +-------------------------------+-------------------------------+
  ~              Closest Trust Point (FQDN)                       ~
  +---------------------------------------------------------------+
```
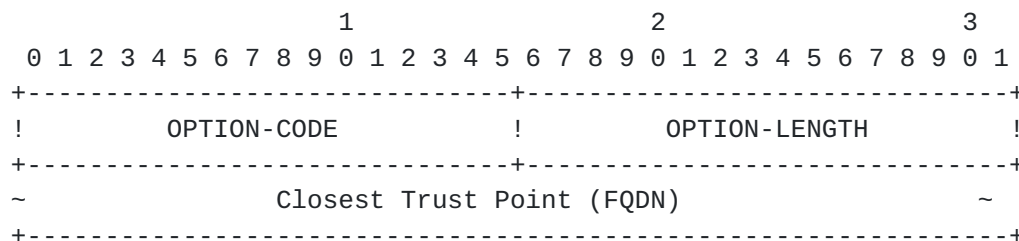
o OPTION-CODE, 2 octets, for CHAIN is 13.

o OPTION-LENGTH, 2 octets, contains the length of the payload
   (everything after Option-length) in octets.

o Closest Trust Point, a variable length Fully Qualified Domain Name
   ("FQDN") in DNS wire format of the requested start point of the
   chain. This entry is the 'lowest' known entry in the DNS chain
   known by the recursive server seeking a CHAIN answer for which it
   has a validated DS and DNSKEY record. The end point of the chain
   is obtained from the DNS Query Section itself. No DNS name
   compression is allowed for this value.

5. Protocol Description

5.1. Discovery of Support

A Forwarder may include a zero-length CHAIN option in a regular query
over any transport to discover the DNS server capability for CHAIN.
Recursive Resolvers that support and are willing to accept CHAIN
queries over source IP verified transport respond to a zero-length
CHAIN received by including a zero-length CHAIN option in the answer.
If not already using a source IP verified transport, the Forwarder
MAY then switch to a source IP verified transport and start sending
queries with the CHAIN option to request a CHAIN response from the
Recursive Resolver. Examples of source IP verification are the 3-way
TCP handshake and UDP with [EDNS-COOKIE].

5.2. Generate a Query

   In this option value, the Forwarder sets the Closest Trust Point in
   the chain - furthest from the root - that it already has a DNSSEC
   validated (secure or not) answer for in its cache. The upstream
   Recursive Resolver does not need to include any part of the chain
   from the root down to this option's FQDN. A complete example is
   described in Section 9.1.

   The CHAIN option should generally be sent by system Forwarders and
   Resolvers within an application that also perform DNSSEC validation.

5.3. Send the Option

   When CHAIN is available, the downstream Recursive Resolver can adjust
   its query strategy based on the desired queries and its cache
   contents.

   A Forwarder can request the CHAIN option with every outgoing DNS
   query. However, it is RECOMMENDED that Forwarders remember which
   upstream Recursive Resolvers did not return the option (and
   additional data) with their response. The Forwarder SHOULD fallback
   to regular DNS for subsequent queries to those Recursive Resolvers.
   It MAY switch to another Recursive Resolver that does support the
   CHAIN option or try again later to see if the server has become less
   loaded and is now willing to answer with Query Chains. A fallback
   strategy similar to that described in [RFC6891] section 6.2.2 SHOULD
   be employed to avoid persistent interference due to non-clean paths.

5.4. Generate a Response

   When a query containing a non-zero CHAIN option is received from a
   Forwarder, the upstream Recursive Resolver supporting CHAIN MAY
   respond by confirming that it is returning a CHAIN. To do so, it
   MUST set the CHAIN option to the lowest Trust Point sent as part of
   the chain, with its corresponding OPTION-LENGTH. It extends the
   Authority Section in the DNS answer packet with the DNS RRsets
   required for validating the answer. The DNS RRsets added start with
   the first chain element below the received Closest Trust Point up to
   and including the NS and DS RRsets that represent the zone cut
   (authoritative servers) of the QNAME. The added RRsets MAY be added
   in matching hierarchical order but a DNS client MUST NOT depend on
   the order of the added RRsets for validation. The actual DNS answer
   to the question in the Query Section is placed in the DNS Answer
   Section identical to the traditional DNS answer. All required DNSSEC
   related records must be added to their appropriate sections. This
   includes records required for proof of non-existence of regular and/
   or wildcard records, such as NSEC or NSEC3 records.

Recursive Resolvers that have not implemented or enabled support for
the CHAIN option, or are otherwise unwilling to perform the
additional work for a Chain Query due to work load, may safely ignore
the option in the incoming queries. Such a server MUST NOT include
an CHAIN option when sending DNS answer replies back, thus indicating
it is not able or willing to support Chain Queries at this time.

Requests with wrongly formatted options (i.e. bogus FQDN) MUST be
rejected and a FORMERR response must be returned to the sender, as
described by [RFC6891].

Requests resulting in chains that the receiving resolver is unwilling
to serve can be rejected by answering the query as a regular DNS
reply but with an empty CHAIN payload. Replying with an empty CHAIN
can be used for chains that would be too big or chains that would
reveal too much information considered private.

At any time, a Recursive Resolver that has determined that it is
running low on resources can refuse CHAIN queries by replying with a
regular DNS reply with an empty CHAIN payload.

If a CHAIN answer would be bigger than the Recursive Resolver is
willing to serve, it SHOULD send a partial chain starting with the
data closest to the top of the chain. The client MAY re-send the
query with an updated Closest Trust Point until it has received the
full chain. The CHAIN response will contain the lowest Closest Trust
Point that was included in the CHAIN answer.

If the DNS request results in an CNAME or DNAME for the Answer
Section, the Recursive Resolver MUST return these records in the
Answer Section similar to regular DNS processing. The CNAME or DNAME
target MAY be placed in the Additional Section only if all supporting
records for DNSSEC validation of the CNAME or DNAME target are also
added to the Authority Section.

The response from a Recursive Resolver to a Resolver MUST NOT contain
the CHAIN option if none was present in the Resolver's original
request.

A DNS query that contains the CHAIN option MUST also have the DNSSEC
OK ("OK") bit set. If this bit is not set, or if the Checking
Disabled ("CD") bit is set, the CHAIN option received MUST be
ignored.

6. Protocol Considerations

6.1. DNSSEC Considerations

The presence or absence of an OPT resource record containing an CHAIN
option in a DNS query does not change the usage of those resource
records and mechanisms used to provide data origin authentication and
data integrity to the DNS, as described in [RFC4033], [RFC4034] and
[RFC4035].

6.2. NS record Considerations

CHAIN responses SHOULD include the NS RRset from the zone itself
including the RRSIG records required for validation. It MUST NOT
include the NS RRset from parent zone, as this RRset is not signed.
If the size of the answer is an important factor, the NS RRset MAY be
omited.

When a DNSSEC chain is supplied via CHAIN, the Forwarder is no longer
required to use the NS RRset, as it can construct the validation path
via the DNSKEY and DS RRsets without using the NS RRset. However,
the Forwarder might be forced to switch from Forwarder mode to
Recursive Resolver mode due to a network topology change. In
Recursive Resolver mode, the NS RRsets are needed to find and query
Authoritative Servers directly. It is RECOMMENDED that the DNS
Forwarder populate its cache with this information to avoid requiring
future queries to obtain any missing NS records. Therefore, CHAIN
responses MUST include the NS RRset from the child zone, including
the RRSIG records required for validation.

6.3. Session Management

The use of [TCP-KEEPALIVE] on DNS TCP sessions is RECOMMENDED, and
thus TCP sessions should not immediately be closed after the DNS
answer to the first query is received.

Both DNS clients and servers are subject to resource constraints
which will limit the extent to which Chain Queries can be executed.
Effective limits for the number of active sessions that can be
maintained on individual clients and servers should be established,
either as configuration options or by interrogation of process limits
imposed by the operating system.

In the event that there is greater demand for Chain Queries than can
be accommodated, DNS servers may stop advertising the CHAIN option in
successive DNS messages. This allows, for example, clients with
other candidate servers to query to establish new sessions with
different servers in expectation that those servers might still allow
Chain Queries.

6.4. Negative Trust Anchors

If a CHAIN answer would intersect with a Negative Trust Anchor
[RFC7646], a partial CHAIN up to the node above the Negative Trust
Anchor should be returned.

6.5. Anycast Considerations

Recursive Resolvers of various types are commonly deployed using
anycast [RFC4786].

Successive DNS transactions between a client and server using UDP
transport may involve responses generated by different anycast nodes,
and the use of anycast in the implementation of a DNS server is
effectively undetectable by the client. The CHAIN option SHOULD NOT
be included in responses using UDP transport from servers provisioned
using anycast unless all anycast server nodes are capable of
processing the CHAIN option.

Since DNS queries using CHAIN may result in longer TCP sessions,
network topology changes may disrupt them more frequently. Anycast
servers MAY make use of TCP multipath [RFC6824] to anchor the server
side of the TCP connection to an unambiguously-unicast address in
order to avoid disruption due to topology changes.

7. Implementation Status

[RFC Editor Note: Please remove this entire seciton prior to
publication as an RFC.]

This section records the status of known implementations of the
protocol defined by this specification at the time of posting of this
Internet-Draft, and is based on a proposal described in [RFC6982].
The description of implementations in this section is intended to
assist the IETF in its decision processes in progressing drafts to
RFCs. Please note that the listing of any individual implementation
here does not imply endorsement by the IETF. Furthermore, no effort
has been spent to verify the information presented here that was
supplied by IETF contributors. This is not intended as, and must not
be construed to be, a catalog of available implementations or their
features. Readers are advised to note that other implementations may
exist.

According to [RFC6982], "this will allow reviewers and working groups
to assign due consideration to documents that have the benefit of
running code, which may serve as evidence of valuable experimentation
and feedback that have made the implemented protocols more mature.
It is up to the individual working groups to use this information as
they see fit".

[While there is some interest, no work has started yet]

8. Security Considerations

8.1. Additional work and bandwidth

Producing CHAIN answers incurs additional load and bandwidth on the
Recursive Resolver. At any time, a Recursive Resolver may decide to
no longer answer with CHAIN answers and fall back to traditional DNS
answers.

8.2. Amplification Attacks

Chain Queries can potentially send very large DNS answers. Attackers
could abuse this using spoofed source IP addresses to inflict large
Distributed Denial of Service attacks using query-chains as an
amplification vector in their attack. While TCP is not vulnerable
for this type of abuse, the UDP protocol is vulnerable to this.

A Recursive Resolver MUST NOT return CHAIN answers to clients over
UDP without source IP address verification. An example of UDP based
source IP address verification is [EDNS-COOKIE]. A Recursive
Resolver refusing a CHAIN option MUST respond with a zero-length
CHAIN option to indicate support for CHAIN queries when a proper
transport is used. It MUST NOT send an RCODE of REFUSED.

8.3. Privacy Considerations

A client producing CHAIN queries reveals a little more information
about its cache contents than regular DNS clients. This could be
used the fingerprint a client across network reconnections. If DNS
privacy is a concern, a CHAIN query client MAY try to use a DNS
transport that provides privacy, such as [DNS-over-TLS] or a trusted
DNS server that is contacted through a VPN connection such as IPsec.

9. Examples

9.1. Simple Query for example.com

o A web browser on a client machine asks the Forwarder running on
   localhost to resolve the A record of "www.example.com." by sending
   a regular DNS UDP query on port 53 to 127.0.0.1.

o The Resolver on the client machine checks its cache, and notices
   it already has a DNSSEC validated entry of "com." in its cache.
   This includes the DNSKEY RRset with its RRSIG records. In other
   words, according to its cache, ".com" is DNSSEC validated as
   "secure" and can be used to continue a DNSSEC validated chain.

o The Resolver on the client opens a TCP connection to its upstream
   Recursive Resolver on port 53. It adds the CHAIN option as
   follows:

   * Option-code, set to 13

   * Option-length, set to 5

   * Closest Trust Point set to "com." (0x03 0x63 0x6f 0x6d 0x00)

o The upstream Recursive Resolver receives a DNS query over TCP with
   the CHAIN Closest Trust Point set to "com.". After accepting the
   query it starts constructing a DNS reply packet.

o The upstream Recursive Resolver performs all the regular work to
   ensure it has all the answers to the query for the A record of
   "www.example.com.". It does so without using the CHAIN option -
   unless it is also configured as a Forwarder. The answer to the
   original DNS question could be the actual A record, the DNSSEC
   proof of non-existence, or an insecure NXDOMAIN response.

o The upstream Recursive Resolver adds the CHAIN option to the DNS
   response as follows:

   * Option-code, set to 13

   * Option-length, set to 5

   * The Closest Trust Point is set to "com." (0x03 0x63 0x6f 0x6d
      0x00)

o The upstream Recursive Resolver constructs the DNS Authority
   Section and fills it (in any order) with:

   * The DS RRset for "example.com." and its corresponding RRSIGs
      (made by the "com." DNSKEY(s))

   * The DNSKEY RRset for "example.com." and its corresponding
      RRSIGs (made by the "example.com" DNSKEY(s))

   * The authoritative NS RRset for "example.com." and its
      corresponding RRSIGs (from the child zone)

   If the answer does not exist, and the zone uses DNSSEC, it also
   adds the proof of non-existence, such as NSEC or NSEC3 records, to
   the Authority Section.

o The upstream Recursive Resolver constructs the DNS Answer
   Section and fills it with:

   * The A record of "www.example.com." and its corresponding RRSIGs

   If the answer does not exist (NODATA or NXDOMAIN), the Answer
   Section remains empty. For the NXDOMAIN case, the RCode of the
   DNS answer packet is set to NXDOMAIN. Otherwise it remains
   NOERROR.

o The upstream Recursive Resolver returns the DNS answer over the
   existing TCP connection. When all data is sent, it SHOULD keep
   the TCP connection open to allow for additional incoming DNS
   queries - provided it has enough resources to do so.

o The Resolver on the client receives the DNS answer. It processes
   the Authority Section and the Answer Section and places the
   information in its local cache. It ensures that no data is
   accepted into the cache without having proper DNSSEC validation.
   It MAY do so by looping over the entries in the Authority and
   Answer Sections. When an entry is validated for its cache, it is
   removed from the processing list. If an entry cannot be validated
   it is left in the process list. When the end of the list is
   reached, the list is processed again until either all entries are
   placed in the cache, or the remaining items cannot be placed in
   the cache due to lack of validation. Those entries are then
   discarded.

o If the cache contains a valid answer to the application's query,
   this answer is returned to the application via a regular DNS
   answer packet. This packet MUST NOT contain an CHAIN option. If
   no valid answer can be returned, normal error processing is done.
   For example, an NXDOMAIN or an empty Answer Section could be
   returned depending on the error condition.

9.2. Out-of-path Query for example.com

A Recursive Resolver receives a query for the A record for
example.com. It includes the CHAIN option with the following
parameters:

o Option-code, set to 13

o Option-length, set to 14

o The Closest Trust Point set to 'unrelated.ca.' (0x09 0x75 0x6e
   0x72 0x65 0x6c 0x61 0x74 0x65 0x64 0x03 0x63 0x61 0x00)

As there is no chain that leads from "unrelated.ca." to "example.com", the Resolving Nameserver answers with an empty CHAIN specified using:

o Option-code, set to 13

o Option-length, set to 0x00 0x00

o The Closest Trust Point is omitted (zero length)

Note that the regular answer is still present just as it would be for a query that did not specify the CHAIN option.

9.3. Non-existent data

A Recursive Resolver receives a query for the A record for "ipv6.toronto.redhat.ca". It includes the CHAIN option with the following parameters:

o Option-code, set to 13

o Option-length, set to 0x00 0x03

o The Closest Trust Point set to 'ca.'

Using regular UDP queries towards Authoritative Nameservers, it locates the NS RRset for "toronto.redhat.ca.". When querying for the A record it receives a reply with RCODE "NoError" and an empty Answer Section. The Authority Section contains NSEC3 and RRSIG records proving there is no A RRtype for the QNAME "ipv6.toronto.redhat.ca".

The Recursive Resolver constructs a DNS reply with the following CHAIN option parameters:

o Option-code, set to 13

o Option-length, set to 0x00 0x00

o The Closest Trust Point is ommited (zero length)

The RCODE is set to "NoError". The Authority Section is filled in with:

o The DS RRset for "redhat.ca." plus RRSIGs

o The DNSKEY RRset for "redhat.ca." plus RRSIGs

o The NS RRset for "redhat.ca." plus RRSIGs (eg ns[01].redhat.ca)

o The A RRset for "ns0.redhat.ca." and "ns1.redhat.ca." plus RRSIGs

o The DS RRset for "toronto.redhat.ca." plus RRSIGs

o The NS RRset for "toronto.redhat.ca." plus RRSIGs (eg
   ns[01].toronto.redhat.ca)

o The DNSKEY RRset for "toronto.redhat.ca." plus RRSIGs

o The A RRset and/or AAAA RRset for "ns0.toronto.redhat.ca." and
   "ns1.toronto.redhat.ca." plus RRSIGs

o The NSEC record for "ipv6.toronto.redhat.ca." (proves what RRTYPEs
   do exist, does not include A)

o The NSEC record for "toronto.redhat.ca." (proves no wildcard
   exists)

The Answer Section is empty. The RCode is set to NOERROR.

10. IANA Considerations

10.1. EDNS0 option code for CHAIN

IANA has assigned option code 13 in the "DNS EDNS0 Option Codes
(OPT)" registry to CHAIN.

11. Acknowledgements

Andrew Sullivan pointed out that we do not need any new data formats
to support DNS chains. Olafur Gudmundsson ensured the RRsets are
returned in the proper Sections. Thanks to Tim Wicinski for his
thorough review.

12. Normative References

[DNS-over-TLS]
          Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D.,
          and P. Hoffman, "DNS over TLS: Initiation and Performance
          Considerations", draft-ietf-dprive-dns-over-tls-05 (work
          in progress), January 2016.

[EDNS-COOKIE]
          Eastlake, Donald., "Domain Name System (DNS) Cookies",
          draft-ietf-dnsop-cookies (work in progress), January 2016.

   [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
              STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987,
              <http://www.rfc-editor.org/info/rfc1034>.

   [RFC1035] Mockapetris, P., "Domain names - implementation and
              specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
              November 1987, <http://www.rfc-editor.org/info/rfc1035>.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/
              RFC2119, March 1997,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "DNS Security Introduction and Requirements", RFC
              4033, DOI 10.17487/RFC4033, March 2005,
              <http://www.rfc-editor.org/info/rfc4033>.

   [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "Resource Records for the DNS Security Extensions",
              RFC 4034, DOI 10.17487/RFC4034, March 2005,
              <http://www.rfc-editor.org/info/rfc4034>.

   [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "Protocol Modifications for the DNS Security
              Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005,
              <http://www.rfc-editor.org/info/rfc4035>.

   [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast
              Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786,
              December 2006, <http://www.rfc-editor.org/info/rfc4786>.

   [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,
              "TCP Extensions for Multipath Operation with Multiple
              Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013,
              <http://www.rfc-editor.org/info/rfc6824>.

   [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms
              for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/
              RFC6891, April 2013,
              <http://www.rfc-editor.org/info/rfc6891>.

   [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running
              Code: The Implementation Status Section", RFC 6982, DOI
              10.17487/RFC6982, July 2013,
              <http://www.rfc-editor.org/info/rfc6982>.

   [RFC7646] Ebersman, P., Kumari, W., Griffiths, C., Livingood, J.,
              and R. Weber, "Definition and Use of DNSSEC Negative Trust
              Anchors", RFC 7646, DOI 10.17487/RFC7646, September 2015,
              <http://www.rfc-editor.org/info/rfc7646>.

   [RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS
              Terminology", RFC 7719, DOI 10.17487/RFC7719, December
              2015, <http://www.rfc-editor.org/info/rfc7719>.

   [TCP-KEEPALIVE]
              Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The
              edns-tcp-keepalive EDNS0 Option", draft-ietf-dnsop-edns-
              tcp-keepalive-05 (work in progress), January 2016.

Author's Address

   Paul Wouters
   Red Hat

   Email: pwouters@redhat.com